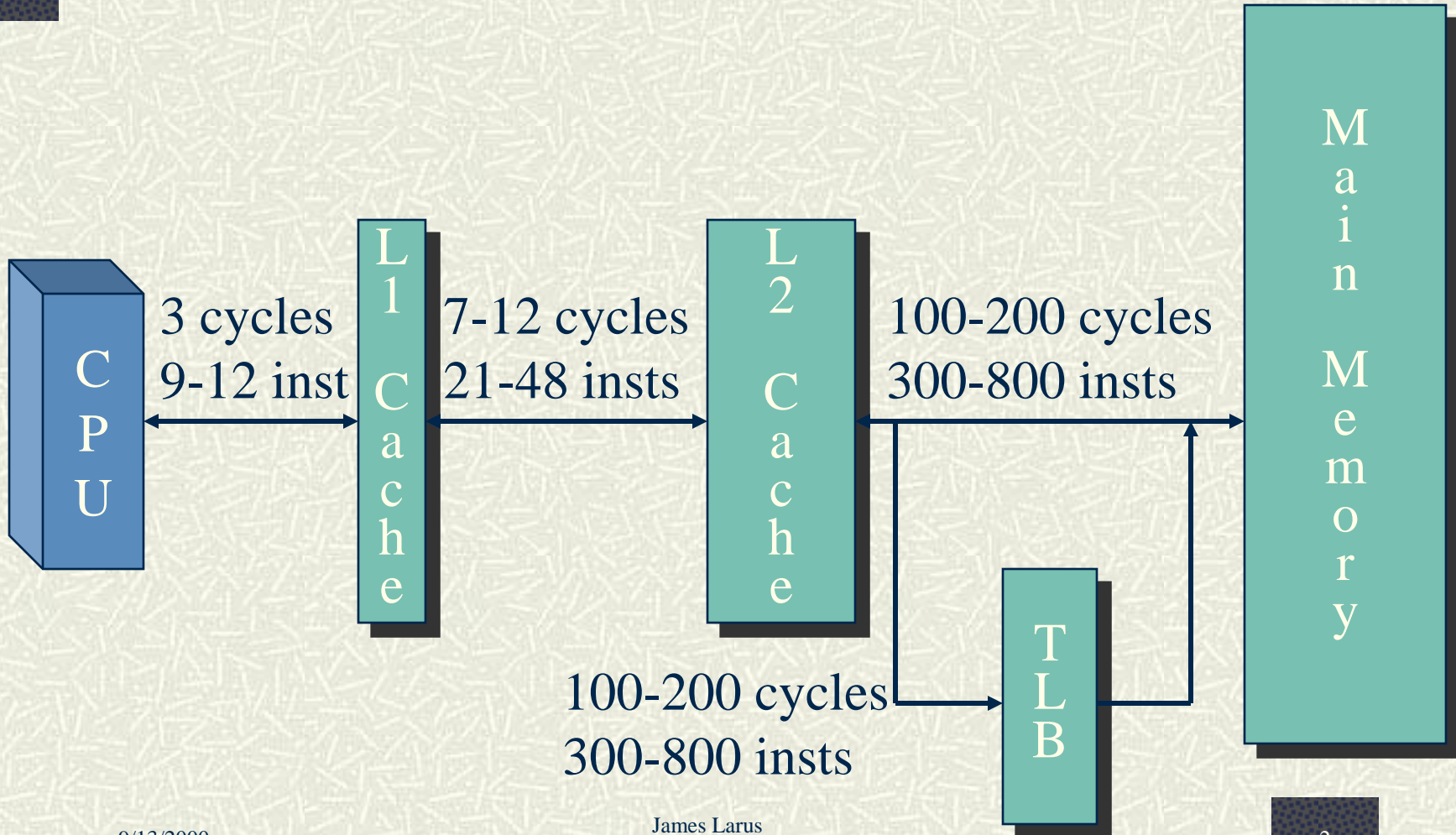


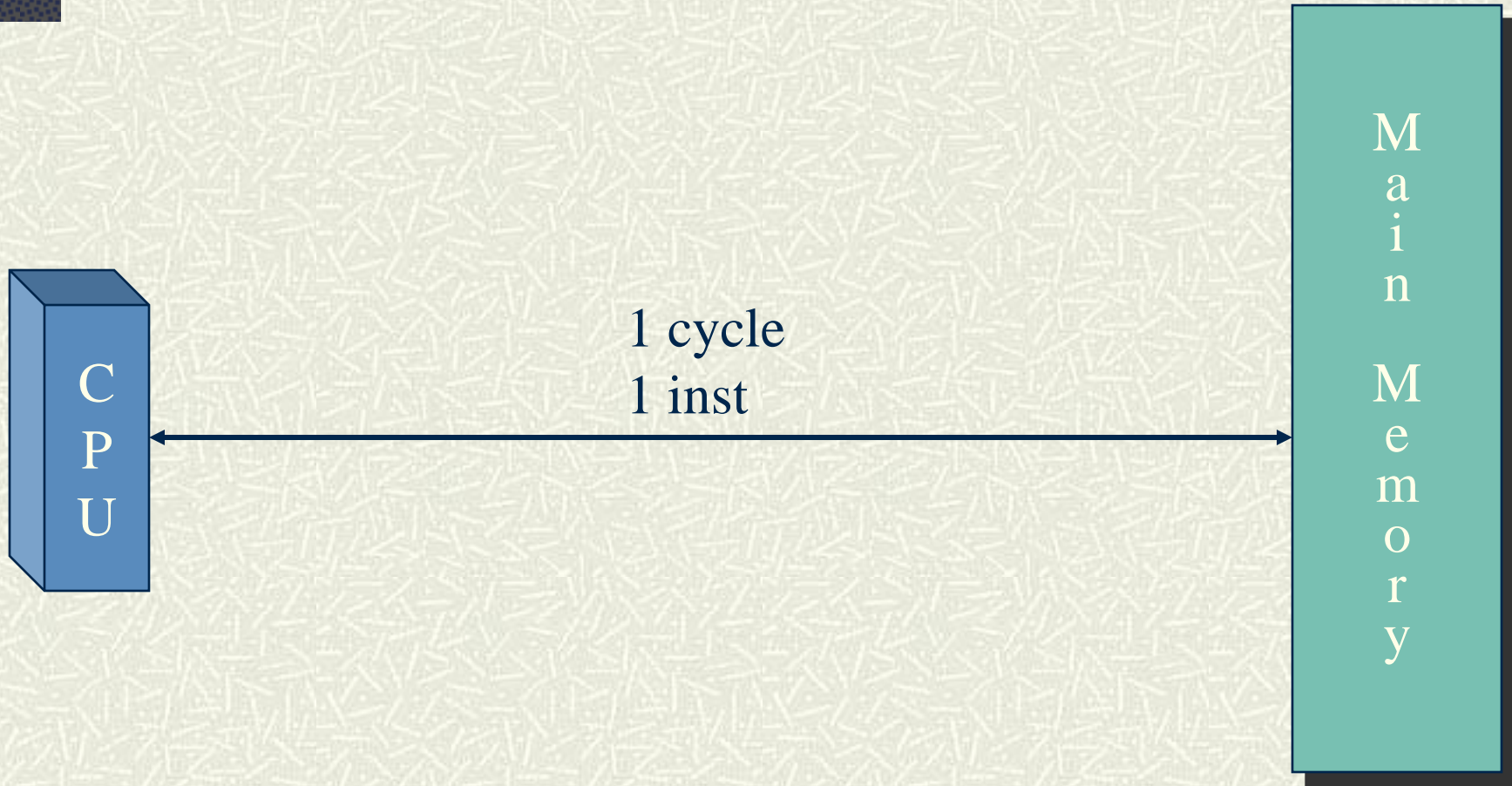
Cache-Conscious Compilation: Can Compilers Hack It?

James R. Larus
Microsoft Research
September 2000

Modern Memory Hierarchy



Programmer's Model



Can Compilers Bridge This Gap?

Current approach

- Schedule loads
 - Separate load & use to hide miss
- Restructure code
 - Blocking/tiling of dense-matrix codes
- Prefetching

Characteristics

- Local changes
- Limited set of programs or limited goals
- Non-semantics preserving \Rightarrow precise program analysis

Another Approach

Cache-conscious programs

- New algorithms
- New data structures
- New program architectures

Change programmer mindset

- New programming model
- Write code to exploit memory hierarchy

Parallel Programming, Redux?

- # Compiler solution to programming problem
 - Limited domains
 - Limited success
- # Should compilers attempt to correct programmer-reality mismatch?

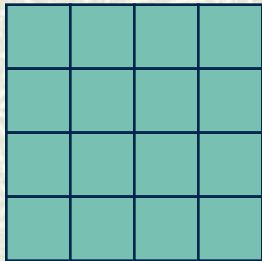
Cache-Conscious Programming

- # Cache-conscious data structures
 - Joint work with Trishul Chilimbi (UW; MSR)
- # Locality-enhanced servers
 - Joint work with Michael Parkes (MSR)
- # Observe:
 - Magnitude of improvement
 - Nature of program changes

Cache-Conscious Data Structures: Key Idea

Arrays

- Random access
- Strong program analysis

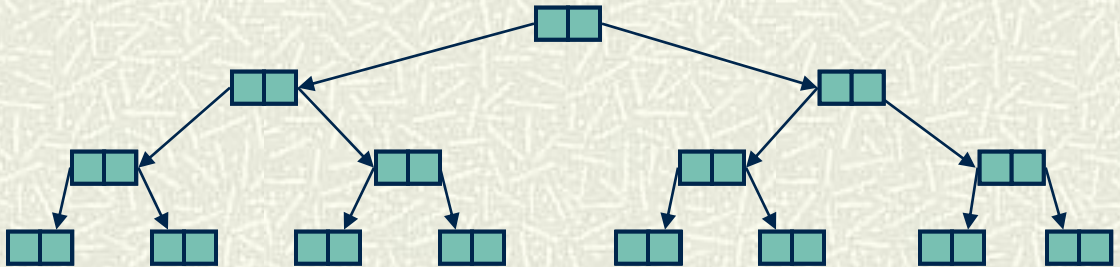


- Data fixed

Change program

Pointer Structures

- Pointer access
- Pointer analysis

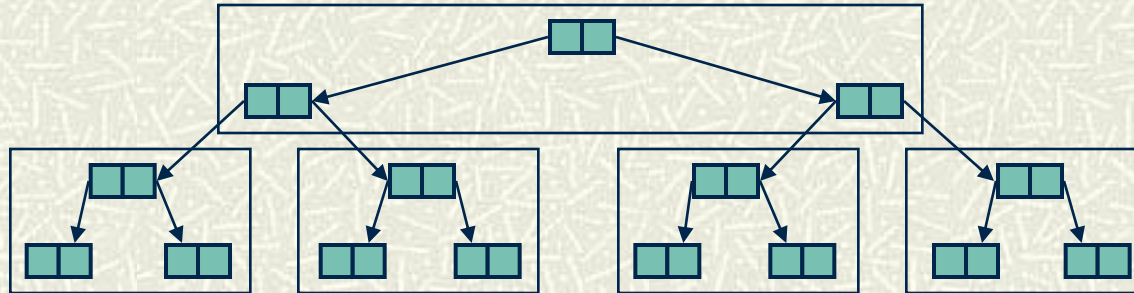


- Data movable

Change data

Technique 1: Clustering

Put contemporaneously accessed items in same cache block



Placement

Expected accesses / block

Random

$$1 + \varepsilon$$

Depth-first

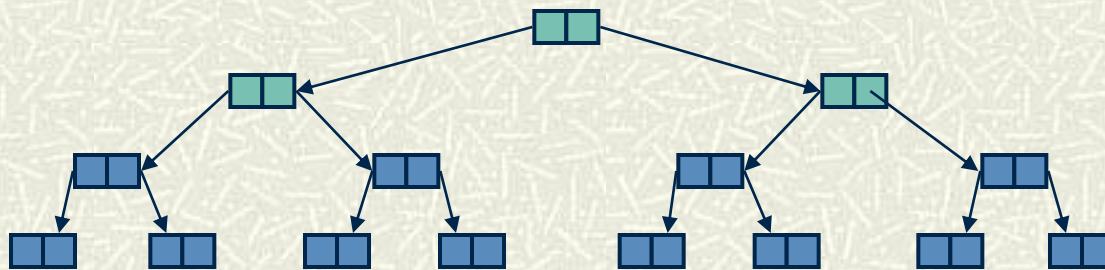
$$2 \times (1 - (0.5)^k)$$

Subtree Clustered

$$\log_2(k + 1)$$

Technique 2: Coloring

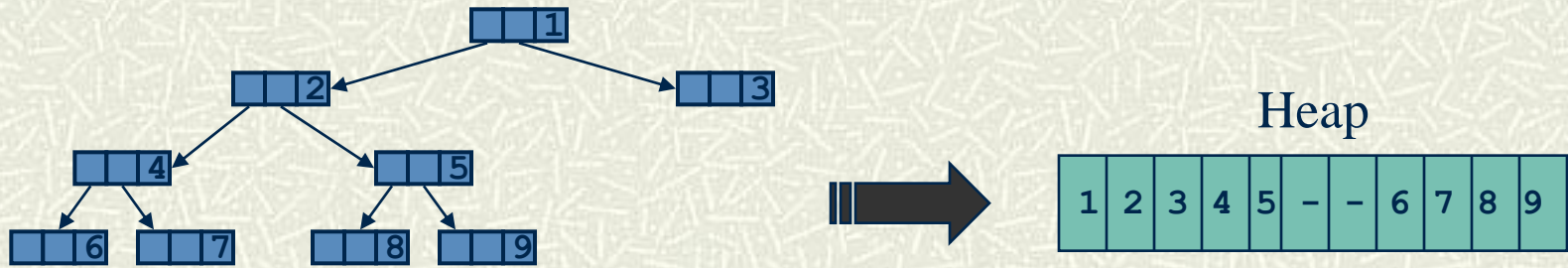
Avoid conflicts for heavily referenced items



Technique 3: Compression

Pack more useful items in cache block

Pointer elimination



Hot/cold splitting



Tools for C

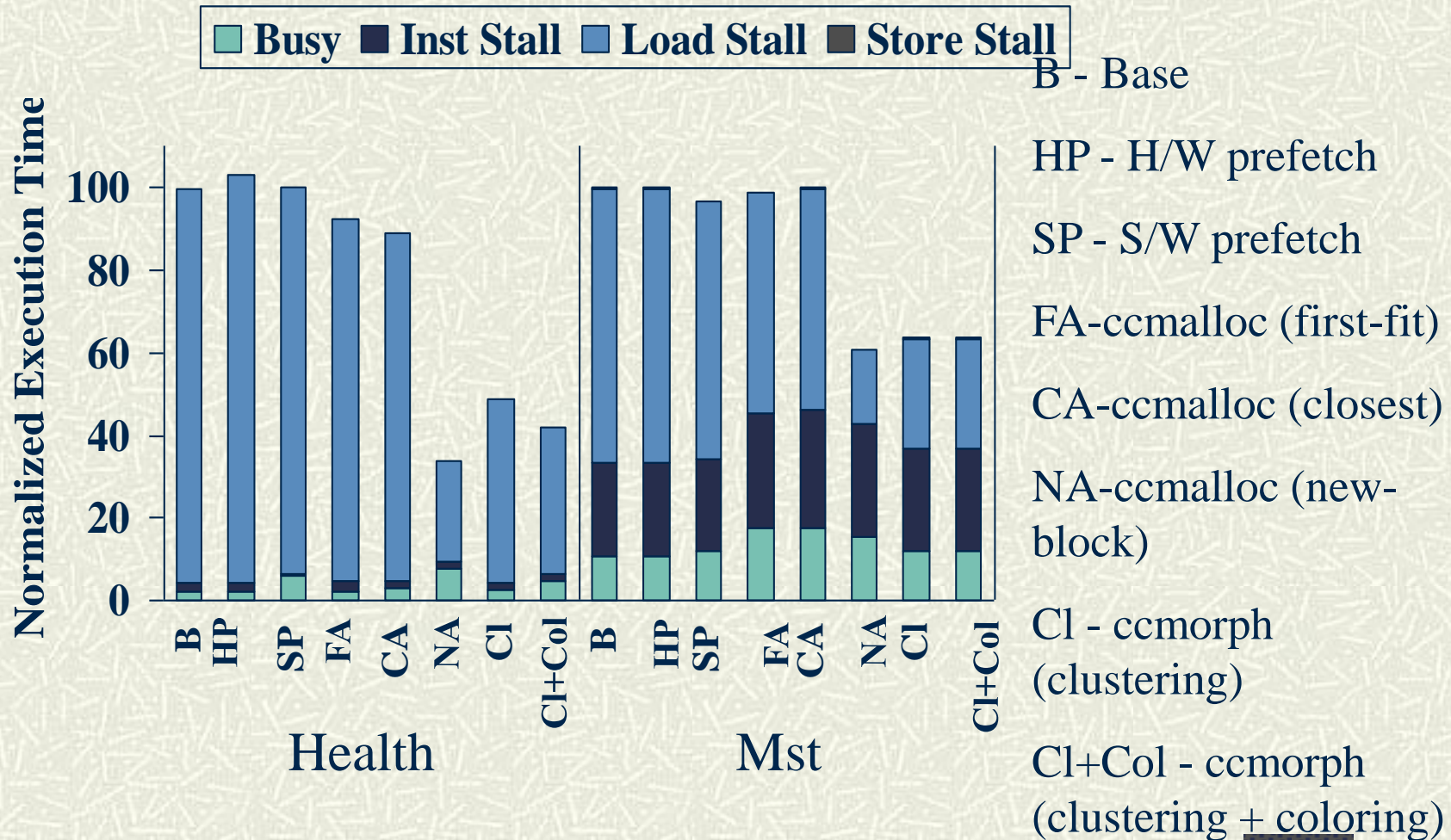
ccmorph

- Topology-based tree reorganizer
- Re-layout tree to increase cache-block reuse

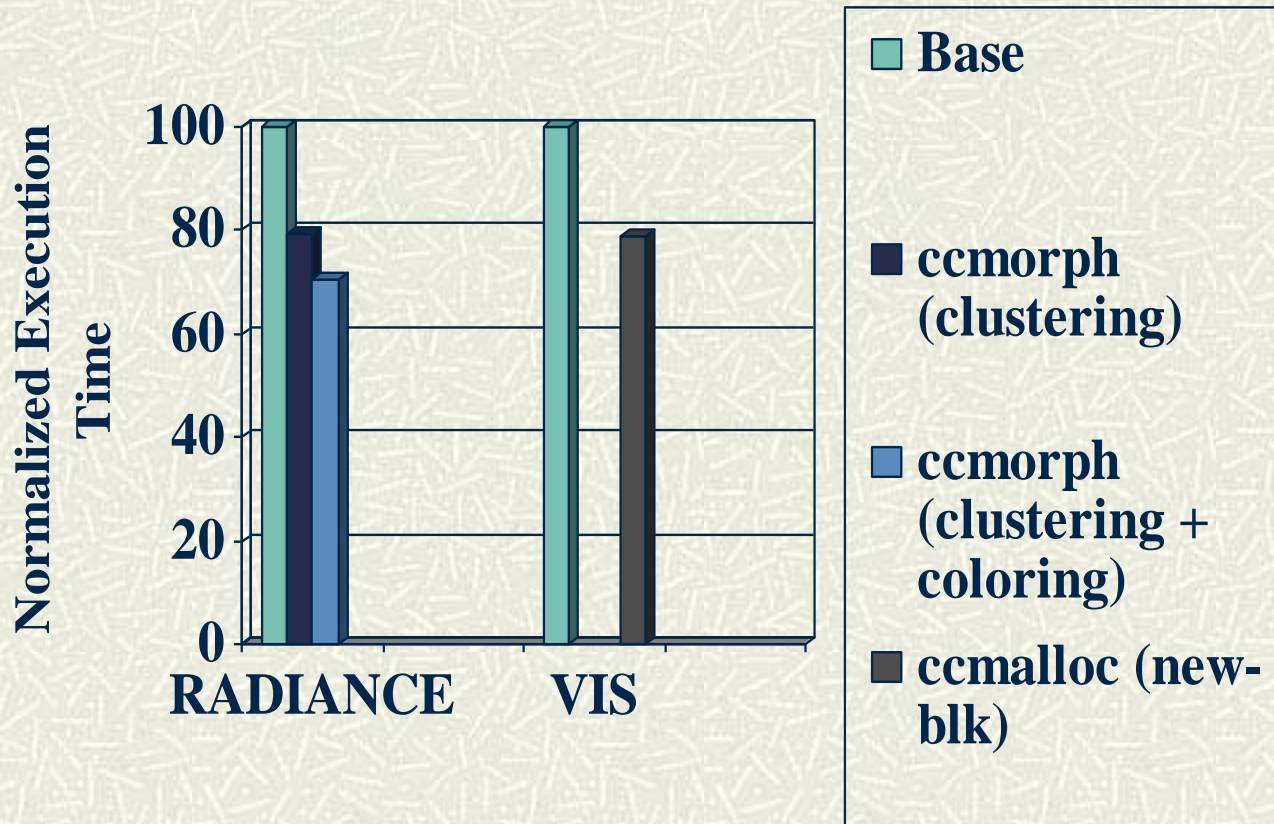
ccmalloc

- Cache-conscious memory allocator
- Specify cache-block affinity with other object

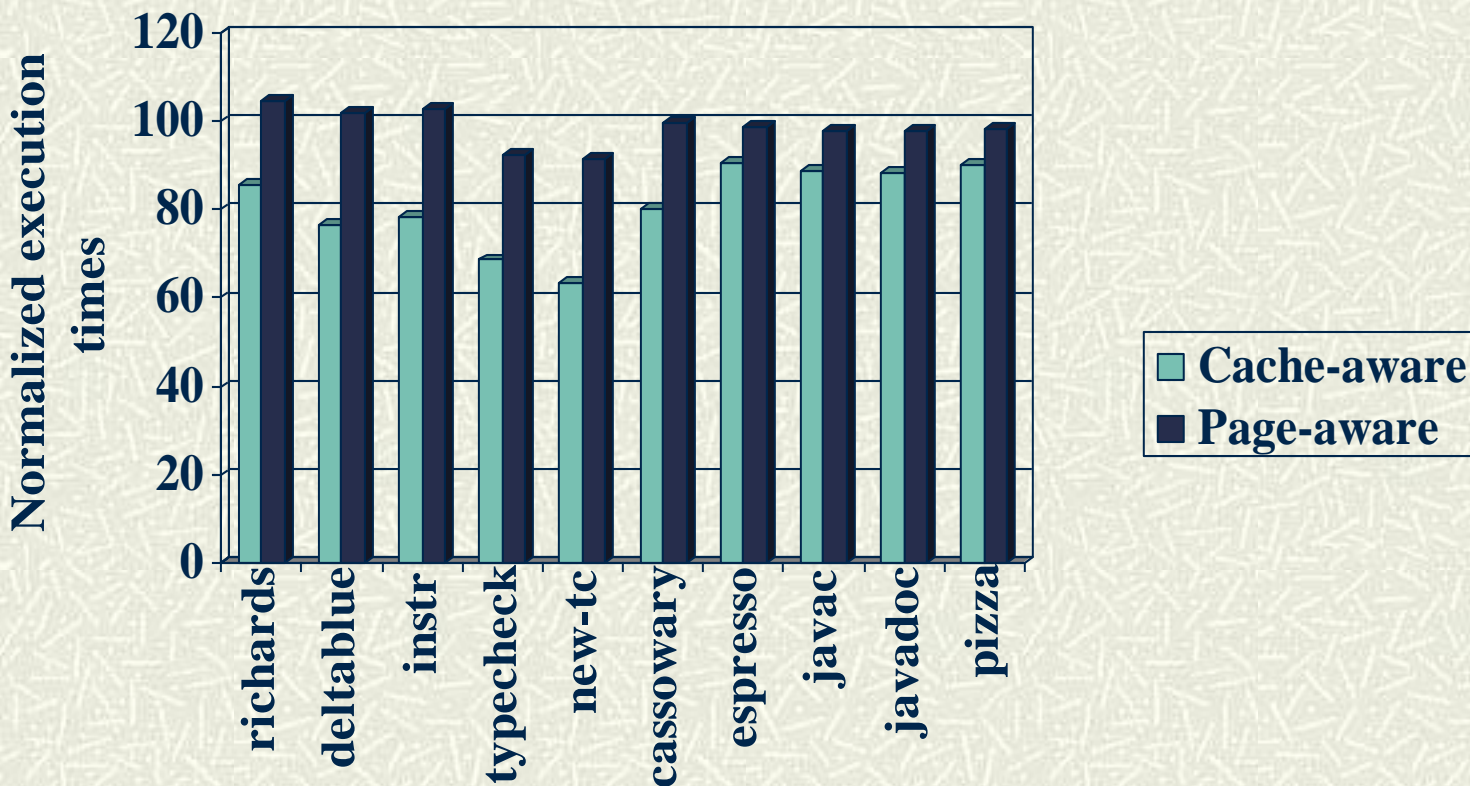
Olden Benchmark Performance



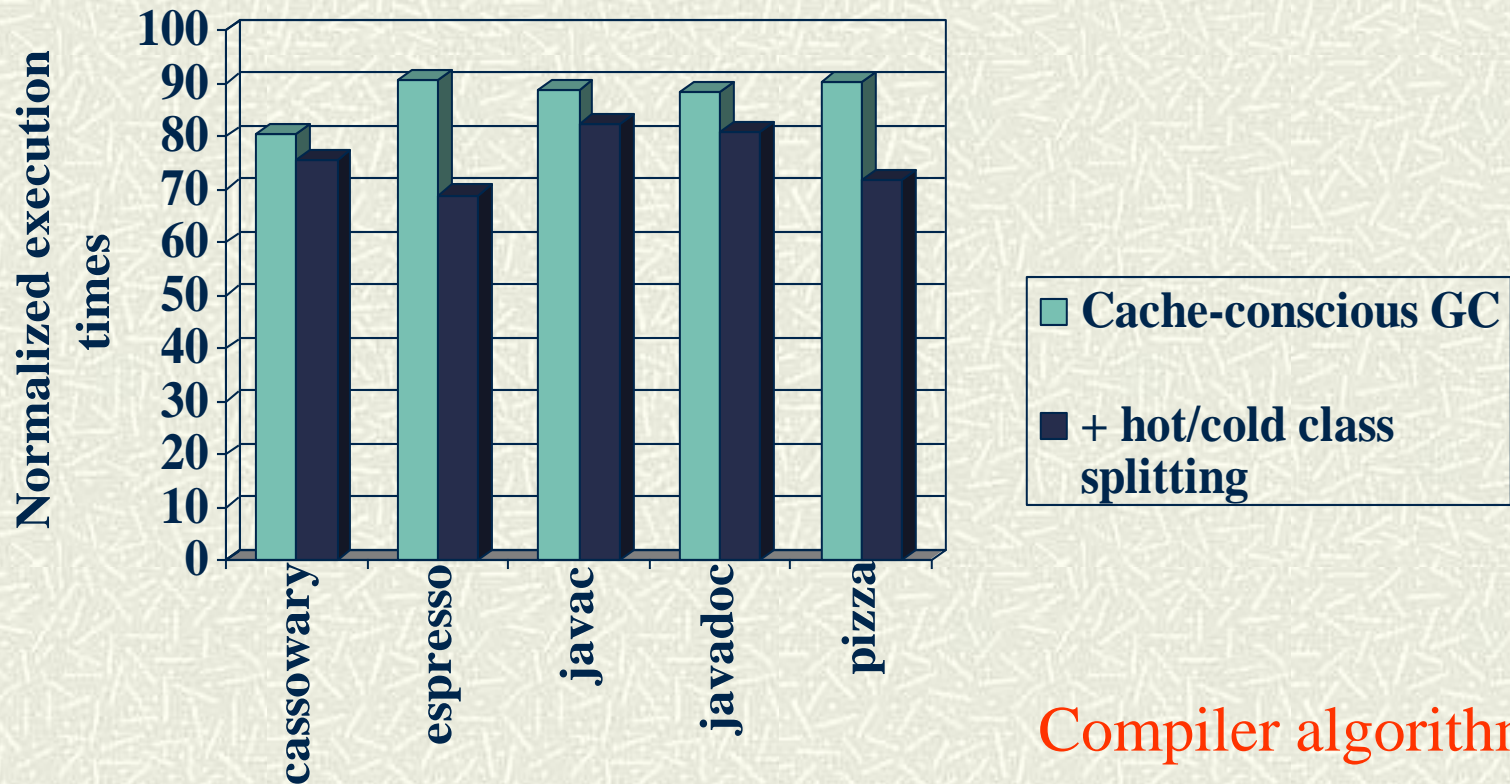
Real Programs



Cache-Conscious Garbage-Collection

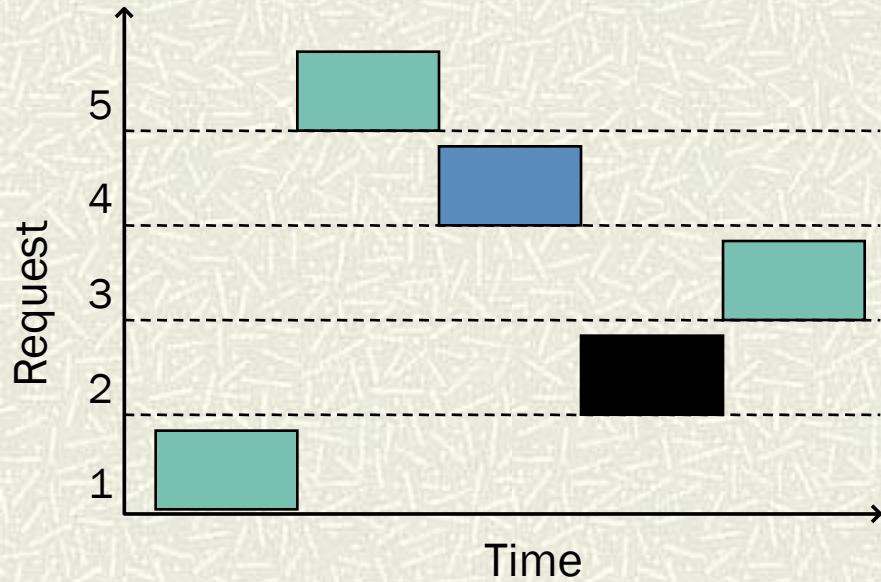


Hot/Cold Class Splitting

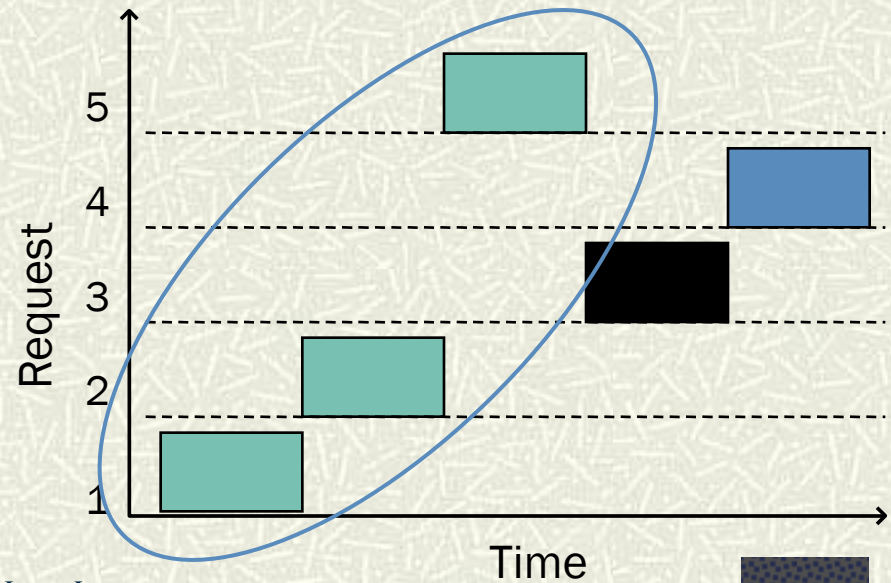


Compiler algorithm!

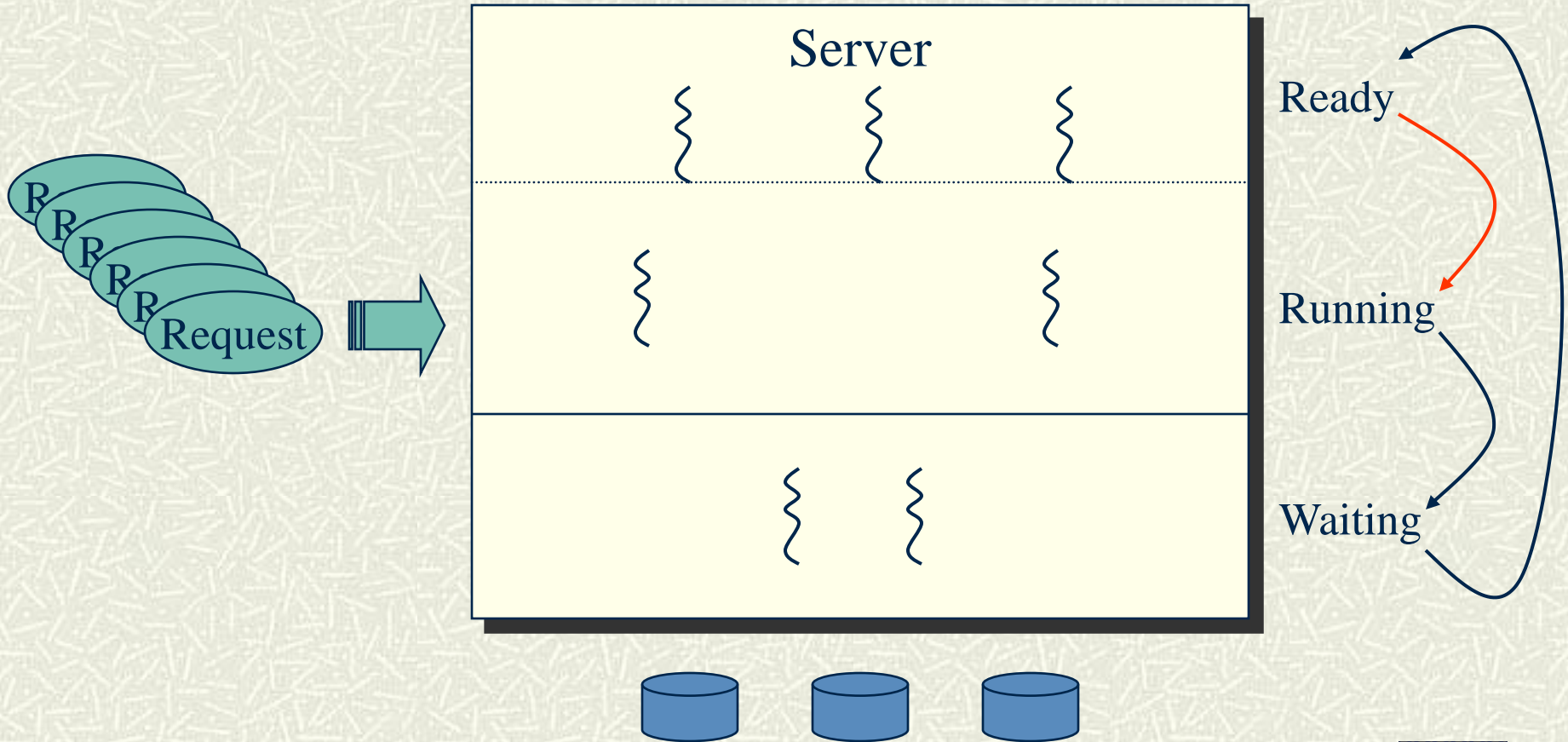
Locality-Enhanced Server: Key Idea



Cohort Scheduling
enhances locality



Threaded Server Architecture



Thread Problems

Poor locality

- Run for short periods (25K inst TPC-C)
- High cost of context switch (for 400K inst)
- Inter-thread conflicts

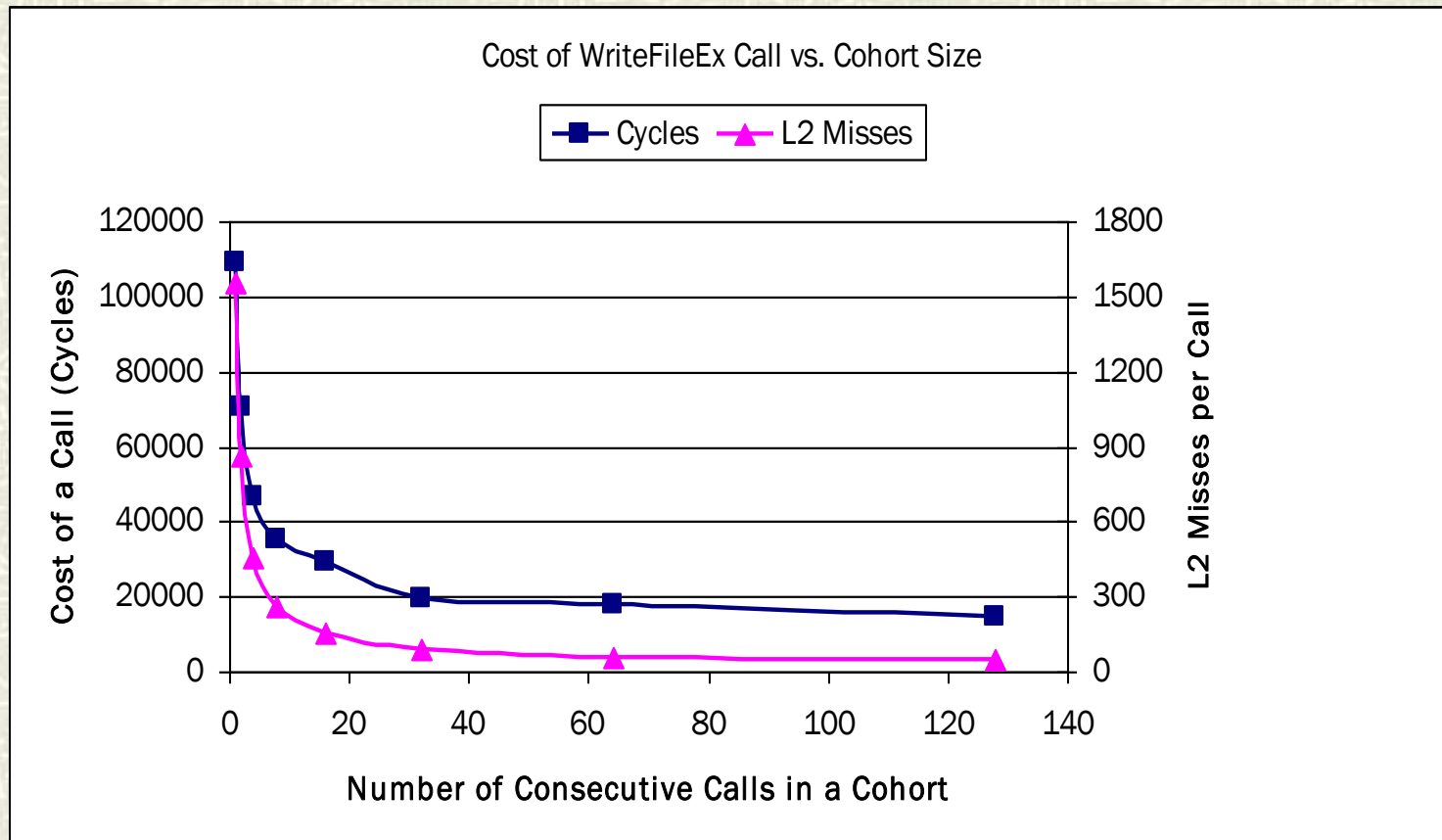
Little locality between threads

- Scheduling has fairness as goal
- Sometimes processor affinity

Cohort Scheduling

- # Aggregate similar operations across server requests
 - Execute consecutively on processor
- # Reduces cold start and conflict misses
 - Improved locality yields higher performance
- # Cache blocking for servers

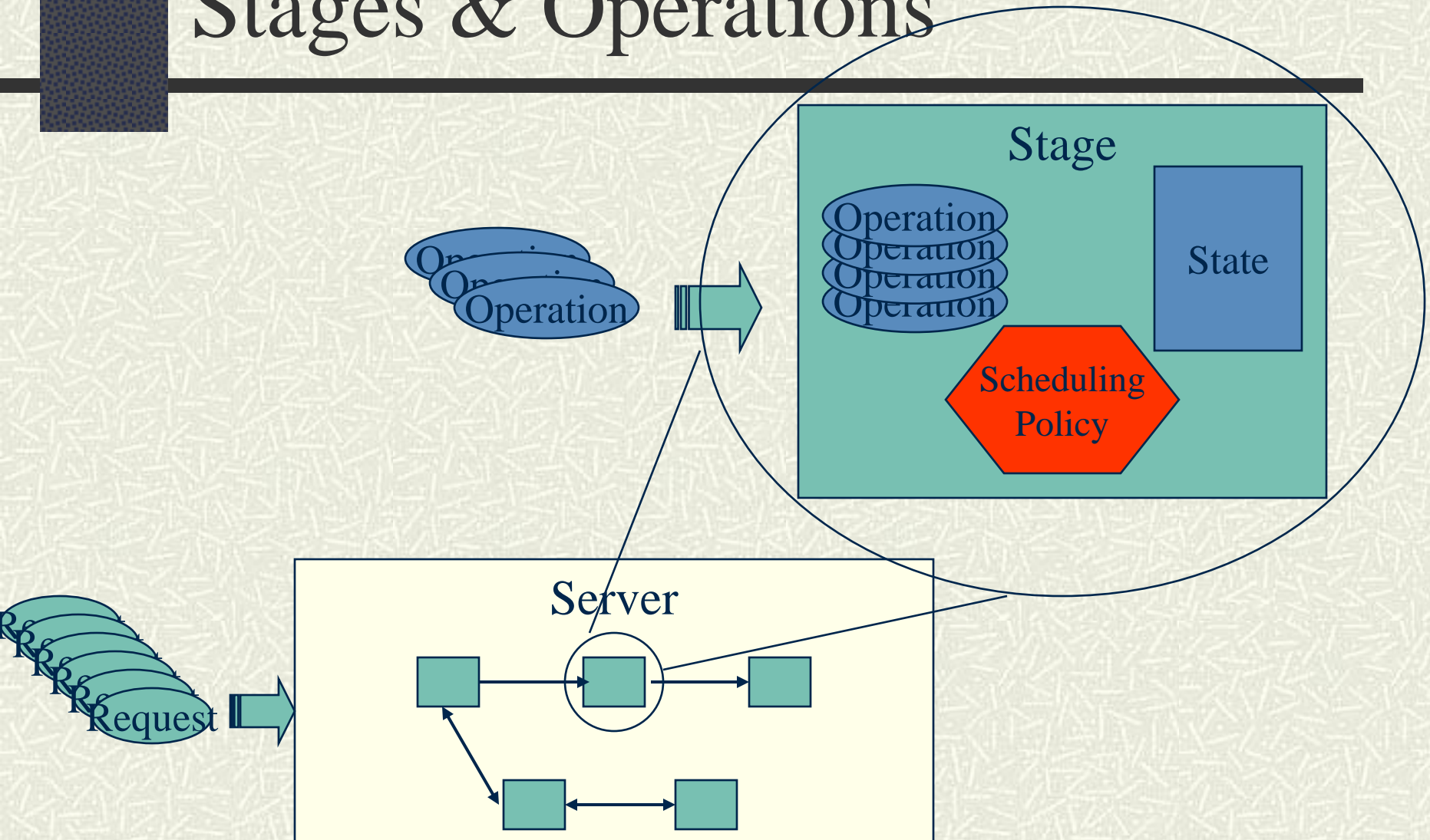
Potential Benefits (OS too)



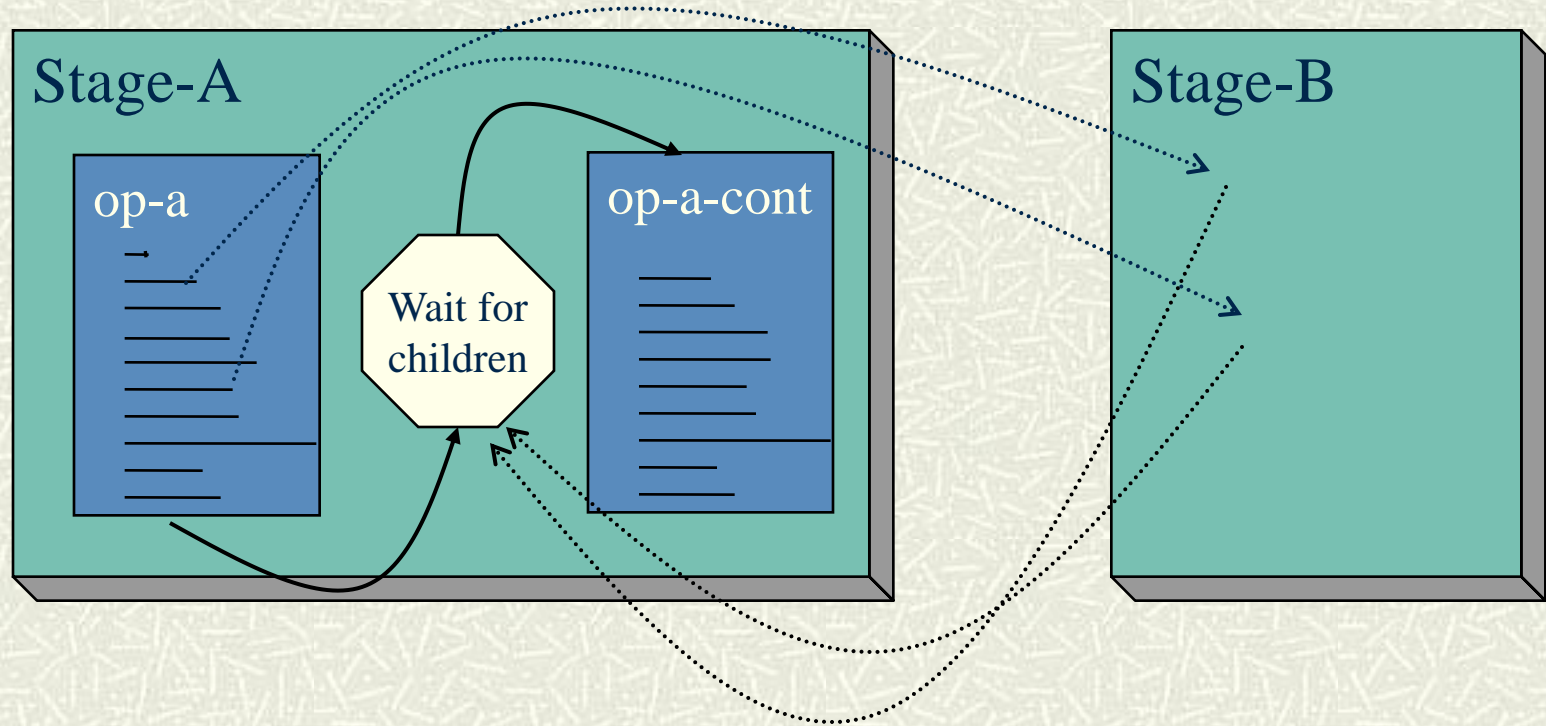
Staged Computation Model

- # Programming model well suited for cohort scheduling
- # Also reduce need for synchronization

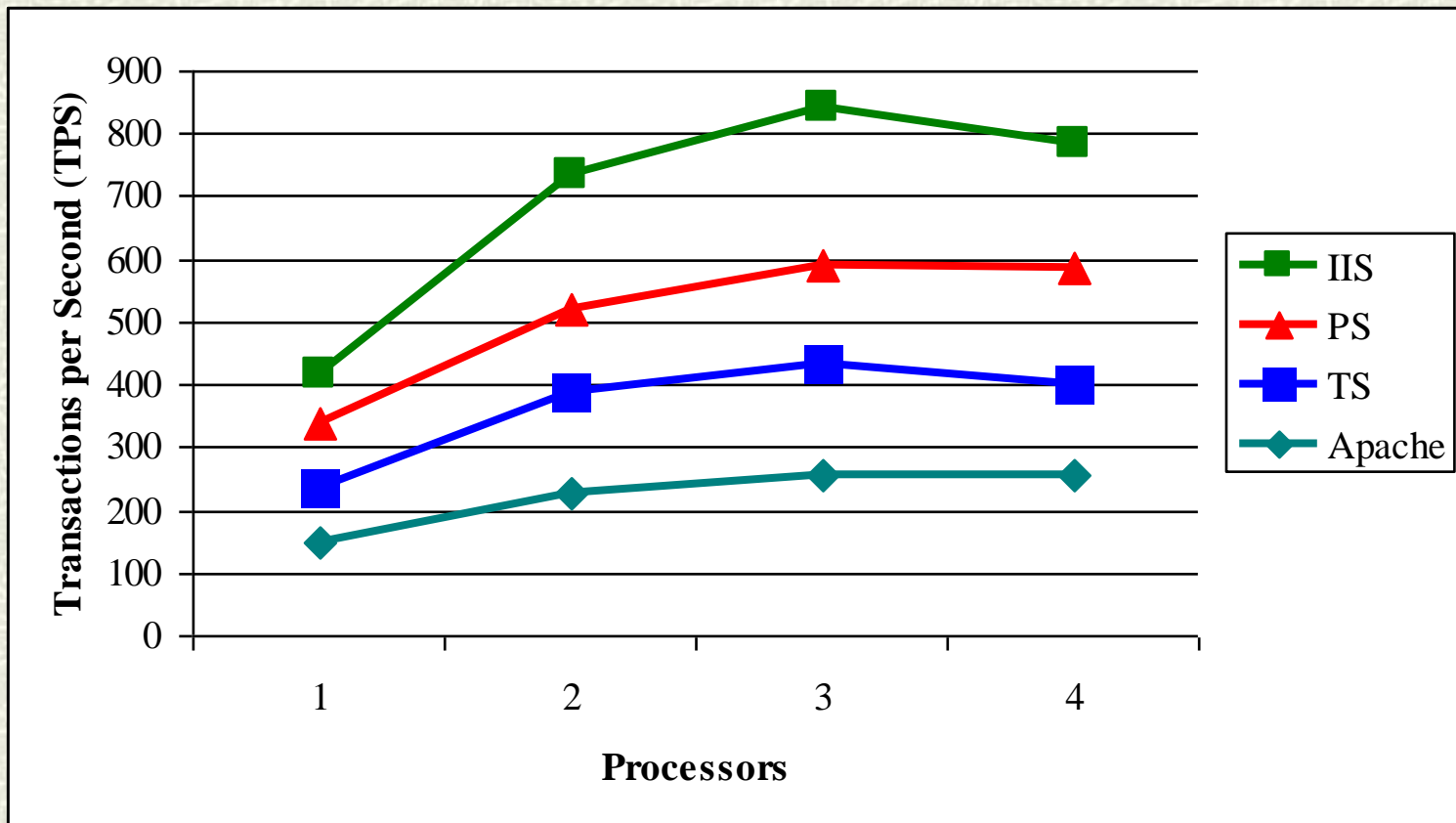
Stages & Operations



Asynchronous Program Model



Web Server Performance



Micro Performance

Average Per Call												
Func.	Inst	PS	TS	TS/PS	PS				TS			
		Time	Time		L1 D	L1 I	L2	Branch	L1 D	L1 I	L2	Branch
		(Cycle)	(Cycle)		(Miss)	(Miss)	(Miss)	(Miss)	(Miss)	(Miss)	(Miss)	(Miss)
sprintf	5335	7099	7920	112%	16	5	10	55	29	12	34	83
strcmp	24	270	315	117%	1	1	1	0	2	2	1	1
time	42	163	461	283%	2	1	2	1	6	4	11	4
tolower	43	53	89	168%	0	0	1	0	1	1	1	1
strncpy	91	101	180	178%	0	1	3	2	1	1	4	4
gmtime	128	453	659	145%	10	2	11	3	14	6	19	8

Could Compilers Replicate These Improvements?

- # Can systematic local improvements fix design flaws?
- # Large-scale program changes
 - Precise whole-program analysis
 - Accurate performance models
- # Change algorithms and data structures
 - How to identify?

Cache-Conscious Compilation?

No: problem is programmers

- Better languages and programming models
- Also education

Compilers can:

- Locally optimize memory use
- Exploit garbage collection
- Avoid unrealistic promises